《C/C++学习指南》

第17.4讲：数据的存储格式

作者：邵发　　QQ群：417024631
官网：http://www.afanihao.cn/c_guide/
答疑：http://www.afanihao.cn/kbase/

数据存储的基本原则

原则：能够写入，也能够读出并还原。（读出指的是数据的解析和还原）
如果写到文件里，却没有办法读出，那就是一个失败的设计

例如，你2个int变量，值为:12345和6790
失败的存储方法：

```
char buf[128];
sprintf("%d%d", a, b);
fwrite(buf, 1, strlen(buf), fp);
```

可行的存储方法:

```
char buf[128];
sprintf("%d,%d", a, b); // 以逗号分隔
fwrite(buf, 1, strlen(buf), fp);
```

如果不以逗号分隔，则无法对数据进行解析。数据白存了！

数据存储的基本原则

常见的文件类型都有自己的存储格式。

bmp

jpg

mp4

mp3

数据的存储格式可以自己定义。只要能满足自己的应用需要就可以了。

存储格式

数据的存储格式是不限的，只要你能满足“能写入、能读出并还原”的原则，哪种方案都可以。

（不考虑方案的优劣）

下面介绍一种最简单的方案：按字节存储

存储格式

按字节存储：所有数据，在内存里的表现都是一串字节，因此，只要将这些字节存入即可。

char / short / int : 占1，2，4个字节
float / double: 4,8个字节
数组:
字符数组:
结构体：
指针：另行讨论。

按字节存储

(1) 基本类型的变量
char/short/int/float/double型变量的存储
只需要知道变量地址和大小

```
// 写入
int a = 0x12345678;
int b = 0x0A0A0A0A;
fwrite(&a, 1, 4, fp); // 4个字节
fwrite(&b, 1, 4, fp); // 4个字节

// 读取
int a ,b;
fread(&a, 1, 4, fp);
fread(&b, 1, 4, fp);
```

按字节存储

(2) 数组的存储
地址：即数组名
字节数：手工计算

```
float arr[4];
fwrite(arr, 1, 4*4 , fp);

// 读取：由于不知道一共存了多个少float，需要循环读取
while(! feof(fp))
{
   float a;
   if(fread(&a, 1, 4, fp) <= 0)
   {
       break;
   }
}
```


按字节存储

(3) 字符数组的存储
（注：这里只是描述一种比较简单的存取方式）

定长方式存取：不论有效长度是多少，统一存储32个字节

```
char text[32];
fwrite(text, 1, 32, fp);

fread(text, 1, 32, fp);
```

按字节存储

(4) 结构体的存储

有两种办法，都比较简单。

第一种办法：直接存取整个结构体。

第二种办法：把每个成员变量依次存储。

```
struct Student
{
    int id;
    char name[16];
    int scores[3];
};
```


按字节存储

整体存取

```
Student s = {201501, "shaofa", {90,90, 90} };

// 写入
fwrite(&s, 1, sizeof(s), fp);

// 读出
fread(&s, 1, sizeof(s), fp);
```

按字节存储

把每个成员变量依次分别存取

```
Student s = {201501, "shaofa", {90,90, 90} };

// 写入
fwrite(&s.id,     1, sizeof(s.id), fp); // int
fwrite(s.name,   1, sizeof(s.name), fp);// char[]
fwrite(s.score, 1, sizeof(s.score), fp); // int[]

// 读出
fread(&s.id,     1, sizeof(s.id),     fp);
fread(s.name,   1, sizeof(s.name),   fp);
fread(s.score, 1, sizeof(s.score), fp);
```


按字节存储

(5) 指针的存储

指针要么不存储，要么存储它指向的对象的内容

（指针本身没必要存储，它只是一个地址）

```
struct Car
{
    char maker[32]; // 制造商
    int  price;  // 价格
};
struct Citizen
{
    char name[32]; // 名字
    int  deposite; // 存款
    Car* car;  // NULL时表示没车
};
```

按字节存储

```
Car* car = (Car*) malloc(sizeof(Car));
strcpy(car->maker, “Chevrolet”);
car->price = 10;

Citizen  who = { "shaofa", 100};
who.car = car;
```

如何存储car的信息？？显示不能存储指针，因为指针只是一个地址。重启程序之后，要能够从文件中还原出Car的信息才行。

按字节存储

写入文件：存储car信息

```
if(who.car != NULL)
{
    fwrite("Y", 1, 1, fp); // 存入一个字节'Y'
    fwrite(who.car->maker, 1, 32, fp);
    fwrite(&who.car->price,1, 4, fp);
}
else
{
    fwrite("N", 1, 1, fp); // 存入一个字节'N'
}
```

按字节存储

从读文件中读出

```
char has = 'N';
fread(&has, 1, 1, fp);
if(has == 'Y') // 先看有没有car的信息
{
    Car* car = (Car*) malloc(sizeof(Car));
    fread(car->maker, 1, 32, fp);
    fread(&car->price, 1, 4, fp);
}
```


小结

介绍一种最简单的存储方式，直接按字节存储。

大家不一定要使用这种方式，但是一定要遵守设计原则：

“能写入，也要能读出并且还原！”